Docket No. 6169-225

METHOD OF CHANGING SERVICE ATTRIBUTES IN A SERVICE LOGIC EXECUTION ENVIRONMENT

Inventors:

Thomas E. Creamer

Zygmunt A. Lozinski

Victor S. Moore

Glen R. Walters

International Business Machines Corporation

IBM DOCKET NO. BOC9-2000-0092

IBM DISCLOSURE NO. BOC8-2000-0058

BACKGROUND OF THE INVENTION

Technical Field

This invention relates to the field of telecommunications, and more particularly, to a method of directly changing service attributes corresponding to service components.

Description of the Related Art

The development of the open network application programming interface (API) represents an important departure from the traditional method by which the architecture of the public switched telephone network (PSTN) was first opened. Traditionally, the Advanced Intelligent Network (AIN) architecture defined a call model which allowed the creation of telecommunications service applications outside of the switch environment. Telecommunications service applications are a la carte telecommunications applications which can perform enhanced services for a telecommunications session established among two or more parties. Exemplary service applications can include Call Waiting, Caller ID, Call Forwarding, Voice Activated Dialing, and Meet-me Conferencing.

When AIN first had been introduced, in terms of the service application creation process, the AIN architecture represented an important advance. AIN separated service development from switching, allowing service logic components to be developed more quickly and placed in specialized network elements attached to databases. Switches, in turn, being free from all service logic, could be optimized for speed and efficiency. Still, typical service applications which were developed to the AIN specification were written in specialized languages by specially trained programmers using specialized service creation environments.

Importantly, future telecommunications networks will be characterized by new and evolving network architectures where packet-switched, circuit-switched, and wireless networks are integrated to offer subscribers an array of innovative multimedia, multiparty applications. Equally important, it is expected that the process by which

telecommunications applications are developed will change, and will no longer solely be the domain of the telecommunications network or service application provider. In fact, in order to provide a broad portfolio of novel, compelling applications rapidly, service application providers will increasingly turn to third-party applications developers and software vendors. Thus, application development in the telecommunications domain will become more similar to that of software and information technology in general, with customers reaping the benefits of increased competition, reduced time to market, and rapid leveraging of new technology as it is developed.

To make this vision a reality, the principles of AIN have been discarded in favor of a new service application component development paradigm. Specifically, it has been recognized that future integrated networks must offer application developers a set of standard, open APIs so that applications written for compatibility with one vendor's system can execute in the system of another vendor. In consequence, the cost of applications development can be amortized, reducing the final cost to the customer. Java™ APIs for Integrated Networks (JAIN) fulfills the requirements of the new service application component development paradigm. Presently, JAIN includes standard, open published Java™ APIs for next-generation systems consisting of integrated Internet Protocol (IP) or asynchronous transport mode (ATM) networks, PSTN, and wireless networks. The JAIN APIs include interfaces at the protocol level, for different protocols such as Media Gateway Control Protocol (MGCP), Session Initiation Protocol (SIP), and Transactional Capabilities Application Part (TCAP), as well as protocols residing in the higher layers of the telecommunications protocol stack.

JAIN includes a set of integrated network APIs for the Java™ platform and an environment to build and integrate JAIN components into services or applications that work across PSTN, packet and wireless networks. The JAIN approach integrates wireline, wireless, and packet-based networks by separating service-based logic from network-based logic. Figure 1 illustrates a conventional JAIN implementation. As shown in Figure 1, a conventional JAIN implementation can include a protocol layer 102 which can include interfaces to IP, wireline and wireless signaling protocols. These

protocols can include TCAP, ISUP, INAP, MAP, SIP, MGCP, and H.323. The JAIN implementation also can include a signaling layer 103 which can include interfaces to provide connectivity management and call control. The conventional JAIN implementation also can include an application layer 104 for handling secure network access and other external services. Finally, the conventional JAIN implementation can include a service layer 106 which can include a service creation and carrier grade service logic execution environment (SLEE) 108.

In JAIN, the protocol layer 102 and the signaling layer 103 are based upon a Java™ standardization of specific signaling protocols and provide standardized protocol interfaces in an object model. Additionally, applications and protocol stacks can be interchanged all the while providing a high degree of portability to the applications in the application layer using protocol stacks from different sources. By comparison, the application layer 104 provides a single call model across all supported protocols in the protocol layer 102. Fundamentally, the application layer 104 provides a single state machine for multiparty, multimedia, and multiprotocol sessions for service components in the application layer 104. This state machine is accessible by trusted applications that execute in the application layer 104 through a call control API.

Notably, applications or services executing at the service level 102 can communicate directly with protocol adapters in the SLEE 108. Protocol adapters typically are class methods, callbacks, event or interfaces that encapsulate the underlying resources such as TCAP, MGCP, etc. The underlying resources can be implemented in many programming languages, but a JAIN-conformant protocol product must provide at least the relevant JAIN API. In contrast, an external application or service executing in the application layer 104 does not have to be aware of the underlying resources and can remain oblivious to the fact that some of its session or call legs may be using different protocols.

Service components 112 are the core JAIN components and can execute in the SLEE 108. More particularly, service components 112 are constructed according to a standard component model and, instantiations of component assemblies execute in

coordination with the SLEE 108. Using information regarding the protocol layer 102 which can be incorporated into the SLEE 108, service components 112 can interact with the underlying protocol stacks without having specific knowledge of the protocol stack. Thus, service components 112 can use the call model provided by the signaling layer to implement telephony services. More importantly, the SLEE 108 can relieve the service components 112 of conventional lifecycle responsibilities by providing portable support for transactions, persistence, load balancing, security, and object and connection instance pooling. In this way, the service components 112 can focus on providing telephony services.

Presently, a user can change various aspects of a service such as call forwarding, call blocking, or messaging, through a Web-based interface such as a hypermedia document or HTML Web page running over the Internet. These service aspects can include service information such as telephone numbers to be blocked, times which particular numbers can be blocked, as well as other behavioral aspects of a service such as the type of message to be played to particular calling numbers or the number of rings before the messaging service answers the telephone. The service information and behavioral parameters of the service collectively can be referred to as service attributes and are conventionally stored within a separate database external to both the hypermedia document and the actual services which can be implemented with one or more service components 112. Once the user initiates changes to the service attributes from the hypermedia document, the service attribute information can be updated in a manner consistent with the user's desired changes.

As shown in Figure 1, conventional systems for updating service attribute information typically are separate and distinct from the services which actually use the information. For example, once the user has requested a change to a service attribute, a hypermedia document 210 can access a database management component 116 to update the service attribute information in a service attribute database 114. Though the service components can contain service attribute information, the service component must access the service attribute information database 114 and synchronize its service

attribute information with the updated information contained within the service attribute database 114. This synchronization process between the service component 112 and the service attribute database 114 often can require another separate and distinct component or service such as synchronization component 118. Notably, the database management component 116, the synchronization component 118, as well as the service attributes database 114, each operate external to the JSLEE 108.

SUMMARY OF THE INVENTION

The invention disclosed herein concerns a method and a system for providing a Web-based interface for directly changing service attributes and corresponding service attribute information. In particular, the invention disclosed herein can provide a common interface for both subscribers and service provider personnel for updating preferences of a subscriber's service. The service attributes and the service attribute information can be contained within a service component executing within a service logic execution environment (SLEE).

The invention can provide this functionality using a hypermedia document in conjunction with the service component without the aid of external applications or services. The hypermedia document can generate a SLEE compatible event responsive to a user request to alter or change service attributes of a service component. The service component can be configured to receive events generated by the hypermedia document interface via the SLEE. Upon receiving such an event, the service component can update service attribute information existing within the service component in a manner consistent with the received event. Accordingly, the service component can include the necessary functionality for updating any service attribute information contained therein and need not access an external database or an external component for accessing such an external database.

One aspect of the present invention can include a method of directly changing a service attribute corresponding to a service component through a hypermedia document. The hypermedia document can provide an interface to a SLEE. The method can include providing a plurality of selections embodied in the hypermedia document, wherein the plurality of selections can correspond to the service attribute. A user specified selection can be received in the hypermedia document and a SLEE compatible event can be generated based on the user selection. The event can be of a type which the service component has been registered in the SLEE to receive. The event can be routed to the service component via the SLEE. The service component can process the event and can update service attribute information corresponding to

the service attribute consistent with the event. Also, an acknowledgment event can be received from the service component.

Another embodiment of the invention can include registering the service component with the SLEE for receiving a SLEE compatible event generated by the hypermedia document. The event can be posted to the SLEE and can be received by the service component. The method further can include updating service attribute information corresponding to the service attribute in the service component. The updating of service attribute information can be consistent with the received event. An acknowledgment event can be routed from the service component to a location in a computer communication network containing the hypermedia document.

Another aspect of the invention can include a machine readable storage, having stored thereon a computer program having a plurality of code sections executable by a machine for causing the machine to perform a series of steps. In that case, the steps can include providing a plurality of selections embodied in a hypermedia document, wherein the plurality of selections can correspond to a service attribute. The service attribute can correspond to a service component executing in a SLEE and the hypermedia document can provide an interface to the SLEE. The method also can include receiving a user specified selection in the hypermedia document and generating a SLEE compatible event based on the user selection. The event can be of a type which the service component has been registered in the SLEE to receive. Also, the event can be routed to the service component via the SLEE. The service component can process the event and update service attribute information corresponding to the service attribute consistent with the event. An acknowledgment event can be received from the service component.

Another embodiment of the invention can include registering a service component with a SLEE for receiving a SLEE compatible event generated by a hypermedia document. The hypermedia document can provide an interface to the SLEE. The event, being posted to the SLEE, can be received by a service component. Service attribute information corresponding to the service component executing in the

SLEE can be updated. The updating of service attribute information can be consistent with the received event. Finally, an acknowledgment event can be routed from the service component to a location in a computer communications network containing the hypermedia document.

BRIEF DESCRIPTION OF THE DRAWINGS

There are shown in the drawings embodiments of which are presently preferred, it being understood, however, that the invention is not so limited to the precise arrangements and instrumentalities shown, wherein:

Figure 1 is a schematic representation of an intelligent network architecture configured in accordance with a conventional JAIN implementation known in the prior art.

Figure 2 is a schematic representation of an intelligent network architecture configured in accordance with the inventive arrangements disclosed herein.

Figure 3 is a flow chart illustrating an exemplary method of changing service attributes.

DETAILED DESCRIPTION OF THE INVENTION

The invention disclosed herein concerns a method and a system for providing a Web-based interface for directly changing service attributes and corresponding service attribute information. The service attributes and the service attribute information can be contained within a service component executing within a service logic execution environment (SLEE). In particular, the invention can provide this functionality without the aid of external applications or services. The service component can be configured to receive events generated by the Web-based interface via the SLEE. Thus, upon receiving such an event, the service component can update service attribute information existing within the service component in a manner consistent with the received event. Accordingly, the service component can include the necessary functionality for updating any service attribute information contained therein and need not access an external database or an external component for accessing such an external database.

Figure 2 is a schematic illustration of a JAIN-compliant intelligent network configured in accordance with the inventive arrangements. The JAIN-compliant network can include a protocol layer 201, a signaling layer 203, an application layer 205, and a service layer 207. The application layer 205 can host external third party applications. Typical third party applications can suit mass-market demand for services such as virtual private networks (VPNs), inbound services and unified messaging. External third party applications also can include short-lived and niche applications which can be deployed using un-trusted application space deployment technologies such as database lookup interfaces, downloadable mechanisms, and the Parlay API, as are well known in the art. Notably, external applications can include hypermedia documents 210 which can be contained within a Web server connected to the World Wide Web.

The hypermedia document 210 can include subscriber information such as phone number lists, access codes, time of day or day of week information, and the like. Service execution options such as preference execution orders, precedence,

exceptions, inclusions, and the like can be included as well. The preference execution order can specify the manner in which a service behaves. For example, the preferences for an answering machine service can specify that for particular incoming calls originating from a particular telephone number, a specific or customized message can be played. Thus, designated business associates can hear one message, designated family members another message, while particular friends can hear yet another message. Further, the preferences can specify an ordering of service features. In that case, the preferences can specify that a call should be screened first, a recording then can be offered, and for particular calling numbers, the called subscriber can be paged.

Precedence can specify particular cases wherein the normal ordering of service features can be defeated. For example, if a telephone call originates from a particular telephone number, the normal ordering of features and the behavior of the service can be altered. Exception and inclusion, which are known in the art, generally can refer to various exception conditions.

The hypermedia document 210 further can present a subscriber with current, real-time usage measurements of the individual service execution components as well as near real-time information. This data can be made available to subscribers in addition to historical data such that future decisions can be made based upon preferences or desired results from past service execution. Those skilled in the art will recognize that the hypermedia document 210 can provide a common interface for both subscribers and service provider personnel for changing service attributes of a subscriber's service.

The service layer 207 can include a SLEE server such as a JSLEE Server 200 which can be configured for compatibility with the JAIN specification. The protocol layer 201 can include one or more protocol stacks which can be configured to interact with the service components 112 executing in the JSLEE Server 200 through a signaling layer 203. Notably, the invention is not limited in regard to the number or type of

protocol stacks. Rather, the JSLEE Server 200 can interact with any protocol stack, for example those protocol stacks configured in accordance with the JAIN specification.

The JSLEE Server 200 also can include several lifecycle management functions. In particular, the service components 112 can be properly loaded within the JSLEE Server 200 for execution. The JSLEE Server 200 can identify configuration and loading parameters associated with each service component 112 to be loaded. Subsequently, the JSLEE Server 200 can execute the service components 112 using the identified configuration and loading parameters. Finally, the service components 112 can register with an internal event handling component in the JSLEE Server 200 so that events can be transmitted to and from the service components 112 executing in the JSLEE Server 200.

In operation, the JSLEE Server 200 can transmit and receive events to and from the protocol stacks in the protocol layer 201. More particularly, the events can be transmitted and received in the event handling component included in the JSLEE Server 200. Likewise, service components 112 which are registered with the JSLEE Server can receive protocol stack events directed towards particular ones of the service components 112. More specifically, the event handling component can route received events to service components 112 which have registered with the JSLEE Server 200 to receive such events. The service components 112 further can post protocol stack events to the JSLEE Server 200.

Figure 3 is a flow chart 300 illustrating an exemplary method of changing service attributes from a Web-based interface. The method of the invention can begin in step 310 where one or more service attribute selections can be presented to a user using a conventional hypermedia document such as a Web page contained within a Web server. In particular, the Web page can include a series of predetermined options from which the user can choose. For example, the user can select items from a list or be guided through a step by step process of selecting from various service attributes. Alternatively, the user can be provided a text box or other interface for entering text such as phone numbers, times, dates, number of rings, or other parameters

corresponding to a particular service. After completion of step 310, the method can continue to step 320.

In step 320, the hypermedia document can receive one or more user specified selections. In step 330, the hypermedia document can generate a SLEE compatible event. For example, the hypermedia document can utilize servlet technology to dynamically generate such events responsive to one or more user selections. Once a SLEE compatible event is generated, the event can be routed to the SLEE server via HTTP and TCP/IP with reference to the SLEE server's URL. After completion of step 330, the method can continue to step 340.

In step 340, the event can be posted to the SLEE server. A service component being registered with the SLEE server to receive such events can receive the event. In step 350, the service component can process or translate the received event such that the service attribute information within the service component can be updated according to the parameters within the received event. After completion of step 350, the method can proceed to step 360.

In step 360, the service component optionally can post an acknowledgment to the SLEE for routing to the Web server and hypermedia document contained therein. Accordingly, information contained within the acknowledgment can be presented to the user by dynamically creating a Web page for presenting that information. For example, the Web page can incorporate dynamic web content creation technology such as Java™ Server Pages. In this manner, two-way communications can be established between the SLEE server and the Web server. Notably, the SLEE server can include service components for sending and receiving information to and from servers on the Internet, as well as service components which can contain and change service attributes. As previously mentioned, these service components can communicate with one another via the event handling component of the SLEE server.

The present invention can be realized in hardware, software, or a combination of hardware and software. A method and system for directly changing service attributes through a Web-based interface according to the present invention can be realized in a

centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system - or other apparatus adapted for carrying out the methods described herein is suited. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention also can be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which when loaded in a computer system is able to carry out these methods. Computer program means or computer program in the present context means any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following a) conversion to another language, code or notation; b) reproduction in a different material form.